Università degli Studi di Bologna
Facoltà di Ingegneria

Progettazione di Applicazioni Web T

Esercitazione 8
Hibernate un esercizio completo

Agenda

- **Esercizio guidato completo per farci trovare pronti alla prova d'esame**

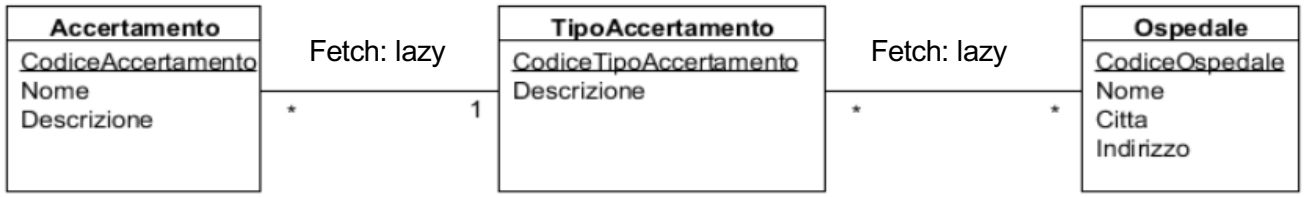
  Progettazione, implementazione e gestione della persistenza basata su metodologia ORM, e in particolare sull'uso della libreira Hibernate, a partire da una realtà di interesse descritta mediante uno schema UML

- Passi principali:
  - Dall'UML ai Java Beans e ai file XML di mapping, dall'UML alle tabelle DB mediate progettazione logica
  - Implementazione delle Relazioni di Associazione in Hibernate
  - Hibernate Config
  - Implementazione delle query

“Accertamenti in ospedale”

N.B. Con un piccolo abuso di notazione, nei diagrammi UML la sottolineatura di un attributo non indica la staticità di tale attributo bensì il suo vincolo di univocita alla E/R.

Partendo dalla realtà illustrata nel **diagramma UML** di seguito riportato, si fornisca una soluzione alla gestione della persistenza basata su **Hibernate** in grado di “mappare” efficientemente e con uso di ID surrogati il modello di dominio rappresentato dai **JavaBean del diagramma UML** con le corrispondenti **tabelle relazionali derivate dalla progettazione logica del diagramma** stesso.



Nel dettaglio, dopo aver creato da applicazione Java gli schemi delle tabelle all’interno del proprio schema nel database **TW_STUD** di DB2 (esplicitando tutti i vincoli derivati dal diagramma UML), implementato i **JavaBean**, definiti i **file XML di mapping** e il **file XML di properties**, si richiede la realizzazione di una classe di prova facente uso delle **API Hibernate** in grado di:

- inserire due o più tuple nelle tabelle di interesse;
- determinare i) elenco dei nomi degli accertamenti di tipo “Analisi di Laboratorio” erogati presso l’ospedale Policlinico S.Orsola-Malpighi di Bologna; ii) per ogni ospedale, nome, città, indirizzo e numero totale di accertamenti erogabili presso la struttura;
- stampare i risultati ottenuti al punto precedente sul file **Ospedale.txt**;

il tutto, mediante opportuna gestione delle **transazioni**.

N.B. La soluzione deve **sfruttare i mapping M-N e 1-N specificati nel diagramma UML**. Ogni ulteriore scelta fatta dallo studente deve essere opportunamente giustificata mediante commenti nel codice.

Primo approccio all'esercizio

Punti salienti:

- Mapping **classi** UML in **Hibernate**
  - tramite **file di mapping XML**
- Mapping delle **relazioni** in **Hibernate** e **verso di percorrenza**
- Risoluzione di query attraverso **Hibernate**

- **Note:** abbiamo già visto cosa vuol dire utilizzare gli ID surrogati, la scrittura su file di testo dei risultati la diamo per assodata

Dall'UML a HIBERNATE: mapping delle Classi

Come sappiamo, il mapping delle classi in Hibernate richiede la definizione di opportuni file XML

- Questi file indicano al framework come "mappare" le entità del dominio (ovvero I nostri Java Beans) nelle corrispondenti tabelle del DB

- Per ogni Java Bean è **necessario** un file XML

Dall'UML a HIBERNATE: mapping delle Classi

Nel dettaglio, per la parte Hibernate:

- Bisogna creare un file XML chiamato:

  <NomeClasseJava>.hbm.xml

- Questo file istruirà il framework Hibernate su come mappare il Java Bean nella tabella corrispondente

- Specificare nome della tabella, nome e tipo degli attributi, chiave primaria, chiavi esterne, vincoli di unicità, ecc.

Dall'UML a HIBERNATE: dal Bean all'XML (1)

Prendiamo come esempio la classe Accertamento…

La parte del Java Bean ormai la diamo per assodata ☺

Oltre agli attributi che ci aspettiamo, si notano due cose:

- L'implementazione dell'interfaccia Serializable (Utile ad Hibernate)
- L'attributo di tipoAccertamento, che dettagliamo in seguito…

```
public class Accertamento implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private int accId;
    private int codiceAcc;
    private String nome;
    private String descrizione;

    private TipoAccertamento tipoAccertamento;
```

Dall'UML a HIBERNATE: dal Bean all'XML (2)

Ora il file **Accertamento.hbm.xml** (i.e., hbm – HiBernate Mapping)

```xml
<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="hibernate.Accertamento" table="accertamenti">
    <id name="accId" column="accId" />
    <property name="codiceAcc" column="codiceAcc"/>
    <property name="nome" column="nome"/>
    <property name="descrizione" column="descrizione"/>

     <many-to-one name="tipoAccertamento" class="hibernate.TipoAccertamento" fetch="select">
            <column name="tipoAccId" not-null="true" />
        </many-to-one>

  </class>
</hibernate-mapping>
```

Dall'UML a HIBERNATE: dal Bean all'XML (3)

Analizziamo il file:

- Hibernate-mapping: Root del file, tutti i file di mapping di Hibernate hanno questo tag come root.
- Class: Specifica il mapping tra la classe Java e la tabella del DB
  - Attributo Name: Nome della classe Java (compresa di package)
  - Attributo Table: Nome della tabella su cui verrà mappata la classe Java
- Id: Indica quale attributo della classe Java farà da chiave primaria della tabella
  - Attributo Name: Nome dell'attributo della classe Java
  - Attributo Column: Nome della colonna della tabella sulla quale verrà mappata la proprietà della classe Java
- Property: Indica il mapping degli attributi comuni
  - Gli attributi sono gli stessi di Id.

Ed i tipi di dato? I vincoli di unicità ? E di not-null? …non è tutto qui!

Dall'UML a HIBERNATE: Dal Bean all'XML (4)

- Dall'esempio mostrato emerge che Hibernate è in grado di derivare i tipi di dato degli attributi delle tabelle da quelli dichiarati nella relativa classe Java
- I vincoli di unicità e di not null sono degli attribute booleani

Esempio completo:

```
<property name="stockName" type="string">
    <column name="STOCK_NAME" length="20" not-null="true" unique="true" />
</property>
<property name="priceChange" type="java.lang.Float">
    <column name="PRICE_CHANGE" precision="6" />
</property>
<property name="volume" type="java.lang.Long">
    <column name="VOLUME" />
</property>
<property name="date" type="date">
    <column name="DATE" length="10" not-null="true" unique="true" />
</property>
```

Abbiamo tralasciato solo una cosa: **Il mapping delle relazioni**!!

HIBERNATE: Il mapping delle Relazioni (1)

Nel diagramma UML presente nel testo **non ci sono frecce per i versi di percorrenza**, sappiamo già che vuol dire:

Relazioni Bidirezionali

- Ecco perchè la classe Java “Accertamento” ha un attributo di tipo “TipoAccertamento”

In Hibernate, dobbiamo mappare la bidirezionalità della relazione tra “Accertamento” e “TipoAccertamento”

La relazione sarà:

- Many-to-One da “Accertamento” verso “TipoAccertamento”
- One-to-Many da “TipoAccertamento” verso “Accertamento”

Queste relazioni dovranno essere dichiarate nei rispettivi file di mapping Hibernate delle classi

- ora vediamo come si fa…

HIBERNATE: Il mapping delle Relazioni (2)

Nel file Accertamento.hbm.xml:

```
<many-to-one name="tipoAccertamento" class="hibernate.TipoAccertamento" fetch="select">
        <column name="tipoAccId" not-null="true" />
    </many-to-one>
```

- Many-to-one: Tipologia di relazione
  - Attributo Name: il nome della proprietà all'interno della classe Java mappata dal file xml in oggetto
  - Attributo Class: Nome della classe Java riferita (esterna)
- Elemento Column: Colonna di riferimento della taballa esterna
  - Attributo Name: Nome della Colonna della primary key della tabella esterna. **N.B. Ovviamente I nomi devono essere uguali!!**

Vedremo un esempio completo...

HIBERNATE: Il mapping delle Relazioni (3)

- Nella classe Java “TipoAccertamento”, essendo una relazione One-to-Many, ci sarà una collezione di oggetti “Accertamento”
- Vediamo come una relazione One-To-Many si mappa in Hibernate, nel file TipoAccertamento.hbm.xml:

```
<set name="accertamenti" table="accertamenti"
            inverse="true" lazy="true" fetch="select">
        <key>
            <column name="tipoAccId" not-null="false" />
        </key>
        <one-to-many class="hibernate.Accertamento" />
    </set>
```

- Set: indica che si sta modellando una molteplicità di oggetti
  - Attributo Name: Nome della proprietà della Collezione nella classe Java
  - Attributo table: Nome della tabella esterna
- Elemento Key: indica la chiave esterna
  - Attributo Name di Column: Nome della Colonna di chiave della tabella stessa
- One-to-many: tipologia della Relazione
  - Attributo Class: Classe Java esterna

HIBERNATE: Il mapping delle Relazioni (4)

Esempio completo:

Accertamento.java

Accertamento.hbm.xml

TipoAccertamento.java

TipoAccertamento.hbm.xml

```java
public class Accertamento implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private int accId;
    private int codiceAcc;
    private String nome;
    private String descrizione;

    private TipoAccertamento tipoAccertamento;
```

```java
private static final long serialVersionUID = 1L;
private int tipoAccId;
private int codiceTipoAcc;
private String  descrizione;

private Set<Ospedale> ospedali = new HashSet<Ospedale>(0);
private Set<Accertamento> accertamenti = new HashSet<Accertamento>(0);
```

```xml
<many-to-one name="tipoAccertamento" class="hibernate.TipoAccertamento" fetch="select">
    <column name="tipoAccId" not-null="true" />
</many-to-one>
```

```xml
<set name="accertamenti" table="accertamenti"
        inverse="true" lazy="true" fetch="select">
    <key>
        <column name="tipoAccId" not-null="false" />
    </key>
    <one-to-many class="hibernate.Accertamento" />
</set>
```

…Abbiamo tralasciato le relazioni Many-to-Many…

HIBERNATE: Relazione Many-To-Many (1)

- "TipoAccertamento" ha anche una relazione Many-to-Many con "Ospedale"
- Come abbiamo visto, nella classe Java "TipoAccertamento" c'è anche una collezione di oggetti "Ospedale"
- Vediamo come questa collezione e questo tipo di relazione si mappa in Hibernate, sempre il file TipoAccertamento.hbm.xml:

```
<set name="ospedali" table="tipoAccertamento_ospedale" inverse="false" lazy="true"
    fetch="select" cascade="all">
        <key column="tipoAccId" />
        <many-to-many column="ospId" class="hibernate.Ospedale"/>
</set>
```

- Il significato degli elementi e degli attributi sono gli stessi della relazione che abbiamo già visto

Attenzione: L'attributo **Table** di **Set** riferisce la **"tabella di mapping"** (ovvero la tabella DB derivata dal passo di progettazione logica applicato alla associazione N-M del diagramma UML di partenza)

Ora vediamo la stessa relazione dalla parte di "Ospedale"…

HIBERNATE: Relazione Many-To-Many (2)

La classe Java “Ospedale”:

```
private int ospId;
private int codice;
private String nome;
private String citta;
private String indirizzo;

private Set<TipoAccertamento> tipiAccertamento = new HashSet<TipoAccertamento>(0);
```

Mentre in Ospedale.hbm.xml:

```
<set name="tipiAccertamento" table="tipoAccertamento_ospedale" inverse="true"
     lazy="true" fetch="select">
    <key column="ospId" />
    <many-to-many column="tipoAccId" class="it.unibo.paw.hibernate.TipoAccertamento" />
</set>
```

Ovviamente è il duale di “TipoAccertamento”

Vediamo ora l’esempio complete…

HIBERNATE: Relazione Many-To-Many (3)

Esempio completo:

TipoAccerttamento.java

TipoAccertamento.hbm.xml

Ospedale.java

Ospedale.hbm.xml

```java
private static final long serialVersionUID = 1L;
private int tipoAccId;
private int codiceTipoAcc;
private String  descrizione;

private Set<Ospedale> ospedali = new HashSet<Ospedale>(0);
private Set<Accertamento> accertamenti = new HashSet<Accertamento>(0);
```

```java
private int ospId;
private int codice;
private String nome;
private String citta;
private String indirizzo;

private Set<TipoAccertamento> tipiAccertamento = new HashSet<TipoAccertamento>(0);
```

```xml
<set name="ospedali" table="tipoAccertamento_ospedale" inverse="false" lazy="true"
    fetch="select" cascade="all">
        <key column="tipoAccId" />
        <many-to-many column="ospId" class="hibernate.Ospedale"/>
</set>
```

```xml
<set name="tipiAccertamento" table="tipoAccertamento_ospedale" inverse="true"
     lazy="true" fetch="select">
    <key column="ospId" />
    <many-to-many column="tipoAccId" class="it.unibo.paw.hibernate.TipoAccertamento" />
</set>
```

Configurare Hibernate: Hibernate Config

…I file xml da scrivere non sono finiti…. L'ultimo file rimanente serve per configurare il framework di Hibernate

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
    <property name="hibernate.connection.username">User</property>
    <property name="hibernate.connection.password">password</property>
  <!--  <property name="connection.pool_size">"1"</property> -->
    <property name="hibernate.dialect">org.hibernate.dialect.DB2Dialect</property>
<!--    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/tw_stud</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password"></property>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>-->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <mapping resource="hibernate/Accertamento.hbm.xml"/>
    <mapping resource="hibernate/Ospedale.hbm.xml"/>
    <mapping resource="hibernate/TipoAccertamento.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Questo file contiene tutti i parametri di configurazione di Hibernate, tra i quali il "dialetto" di SQL che si vuole usare, credenziali d'accesso al DB fisico, e dove si trovano i file di mapping

Il Codice Java e le Query

Una volta impostato e configurato bene il Hibernate framework ed il mapping, le query richieste da questo esercizio non sono di particolare difficoltà:

- L'elenco dei nomi degli accertamenti di tipo "analisi di laboratorio" effettuate al Sant'Orsola di Bologna

- Per ogni Ospedale, restituire: il nome, la città, l'indirizzo, e il numero totale di "Accertamenti" erogabili

Nelle due query che vedremo nelle prossime due slide, è stato utilizzato un **approccio più Java-oriented**, mantenendo le query a livello DB molto semplice… ma una **soluzione DB-oriented** può risultare più efficiente! ☺

…Siete invitati a implementare entrambe le soluzioni!

Prima Query: Le Analisi del Sant'Orsola

Codice soluzione della prima query

```
Query firstQuery = session.createQuery("from "+Ospedale.class.getSimpleName()+
        " where citta = ? and nome = ?");
firstQuery.setString(0, "Bologna");
firstQuery.setString(1, "S.Orsola");
List<Ospedale> ospedaliRes = firstQuery.list();
String firstQueryResult ="";
String secondQueryResult ="";
for(Ospedale os : ospedaliRes)
{
    Set<TipoAccertamento> tipiAcc = os.getTipiAccertamento();
    for(TipoAccertamento tipo : tipiAcc)
    {
        if(tipo.getDescrizione().compareTo("analisi di laboratorio")==0)
        {
            Set<Accertamento> accertamenti = tipo.getAccertamenti();
            for(Accertamento acc :accertamenti)
            {
                firstQueryResult=firstQueryResult+acc.getNome()+"\n";
            }
        }
    }
}
```

Seconda Query: Gli Ospedali ed Accertamenti

Codice soluzione della seconda query

```
Query secondQuery = session.createQuery("from "+Ospedale.class.getSimpleName());
ospedaliRes= secondQuery.list();
for(Ospedale os : ospedaliRes)
{
    secondQueryResult = secondQueryResult+os.getNome()+ " "
            +os.getIndirizzo() +" "+os.getCitta()+"\n Numero Accertamenti erogabili: ";
    Set<TipoAccertamento> tipiAcc = os.getTipiAccertamento();
    int numAccCounter=0;
    for(TipoAccertamento tipo : tipiAcc)
    {
        numAccCounter += tipo.getAccertamenti().size();
    }
    secondQueryResult=secondQueryResult+numAccCounter +"\n";
}
```

Ulteriori link

- **API ufficiali di Hibernate (versione 4.3) al seguente link**
  - http://docs.jboss.org/hibernate/core/4.3/javadocs/

- **Altri link utili**
  - https://www.mkyong.com/tutorials/hibernate-tutorials/
  - https://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/
  - https://www.mkyong.com/hibernate/hibernate-many-to-many-relationship-example/